Description of the files in project source code:

CleanData.ipynb

- Generates new duplicate sentences by using the transitive property of if A == B and B == C, then A == C
- Clean the data
- Creates one CSV file (fulltrain.csv) that contains 3 types of sentences:
  - only cleaned
  - cleaned and removed stopwords
  - cleaned and removed stopwords and lemmatize words

FindClusters.ipynb

- Using the transitive property of if A == B and B == C, then A == C, find the sentences that are clustered together in the cleaned_features dataset by creating graphs

FeatureExtraction.ipynb

- Rebalances the dataset such that there is an equal number of sentences in the duplicate class and the non-duplicate class by discarding some of the generated duplicate sentences
- Extract features for the 3 types of sentences
- Creates a CSV file for each type of sentence for easy machine learning
  - cleaned_features, stopword_features, stopword_lemmatize

TopicModels.ipynb

- Visualise word cloud to find out most commonly occurring words in both questions
- Performs topic modelling to find out the common topics (e.g., health, politics) in question pairs
- Common words and topics were selected and analysed with LIME for logistic regression and random forest model subsequently

LSTM_training.ipynb

- Trains different LSTM models using glove vectors and the cleaned_features dataset
  - Model train: model A that multiples the LSTM output, model B that concatenates the LSTM output, model C which is model A with fewer layers, model D which is the maLSTM
- Evaluates and saves the trained models

LSTM_Model_Inference.ipynb

- Loads a saved LSTM model for new inference/reevaluation

Logr_kNN_train.ipynb
For both cleaned_features, stopwords_lemmatize datasets

- Tried gensim and spacy library to vectorise words - went with spacy lib
- Create features: manhattan_distance, cos_similarity_spacy from word vectors

    - Word vectors are generated from spacy lib and weighted using their tf-idf scores
    - Average of the weighted word vectors of each sentence = sentence vector
    - Calculate the distance (using manhattan distance and cos similarity) between the pair of word vectorised questions in each question set and update the new cols
- Plot correlation matrix for feature selection
- Plot feature importance of all features for logistic regression
- Train top features on logistic regression, using GridSearchCV to get the optimal params
- Compute cos similarity, manhattan distance and horizontally stacked vectorised q1 and vectorised q2. Train these computed vectors on logistic regression using GridSearchCV to get the optimal params.
- Use Lime to explain several instances trained by logistic regression
- Train mean word vectors on kNN using GridSearchCV to get the optimal hyperparameters
- Evaluates the results using macroscopic performance metrics

SVM_Quora Questionpairs.ipynb
For cleaned_features, stopwords_lemmatize and stopwords

- Trained on basic features and features like manhattan distance and cosine similarity were extracted from word2vec embeddings
- Used L1 and L2 regularization to evaluate model performance on linear SVM
- SGD Classifier used for SVM model to minimise loss of our linear classifier
- Calibrated Classifier CV using cross validation to estimate parameters of a classifier and subsequently calibrate our linear classifier.
- Evaluation metrics such as precision, recall , accuracy, F1-score and AUC-ROC obtained

Random_Forest.ipynb
For cleaned_features, stopwords_lemmatize and stopwords

- Hyperparameter tuning using RandomizedSearchCV and GridSearchCV
- Feature engineering with feature importance plots
- Evaluation metrics (precision, recall , accuracy, F1-score and AUC-ROC)
- LIME (true positive, true negative, false positive, false negative)

XGBoost .ipynb

- Trained on basic features and advanced features; features were selected via the correlation matrix
- Conducted cross-validation to select the hyperparameters such as the number of trees and the learning rate
- Performed LIME to understand the microscopic performance of the model
- Evaluate the model with various performance metrics, such as accuracy, precision, recall, F1 score, AUC-ROC and log loss